FH Vorarlberg
University of Applied Sciences

AdaSOTL - An Adaptive Self-Organising Traffic Light control approach

Technical Report
for the course(s)

Agent Based Computer Simulation | Simulation and Optimization

3. Semester
Master Informatics/Mechatronics - WS 2021/22

Marcus Bentele
Georg Fessler
Thomas Rosenberger

FH Vorarlberg – University of Applied Sciences

Dornbirn | February 22, 2022

Abstract

This technical report explores different approaches to traffic light control. These are classical fixed time, cycle based and the decentralised, agent-based control, which are compared against each other on selected networks.
For this comparison their performance on different selected scenarios is examined. For optimization and the following experiments the open-source microscopic road traffic simulator, *Simulation of Urban Mobility* (SUMO) as simulation environment is employed. The results of the algorithms are presented on two selected dynamic traffic scenarios and one scenario with a constant flow of vehicles. To measure performance the average speed and the average waiting time of the vehicles for the different networks are compared. The gained results indicate that the decentralized approach, especially the proposed AdaSOTL as well as the PBSS are the best performing algorithms. However in the network where the flow remains constant the cycle based approach performs solid as well.
Thus, the fixed-time, cycle based control approach is not as effective in networks where the flow changes. This means that this control method has a hard time to adapt to macroscopic changes. In those scenarios decentralised, agent-based methods perform better. However, some further testing of the proposed AdaSOTL Algorithm on other networks is needed to show that it can outclass the PBSS Algorithm.

Contents

List of Figures

List of Tables

Listings

1 Introduction

As part of the courses *Agent Based Computer Simulation* and *Simulation and Optimization* in the winter semester 2021/2022, we dealt with the topic of implementing traffic light controls as part of a project work. This project work is described in more detail in the following technical report. Hence, the main focus of this report is on the methodological approach and the documentation of the results achieved within the semester.

A recent study on urban mobility in the European Union [5] indicates that there is no clear trend towards more sustainable modes of transport. Although cities within the European Union have put a range of initiatives in place to expand the quality and quantity of public transport, overall there has been no significant reduction in private car usage. However, within the last years during the pandemic and more home-office work, these numbers plummeted somewhat.

Nevertheless, it is generally recognised that improved traffic signal control offers the biggest pay-off for reducing congestion on streets, and that signal control systems that adjust to the current traffic conditions (as opposed to more conventional, fixed signal timing systems) offer the most potential. Unfortunately, such adaptive traffic signal control are harder to implement in practice because they need to respond efficiently and effectively to real-time traffic demand changes.

However, simulations are often used to test new systems before implementing them into the real world. This also counts for traffic simulations. Therefore, simulation is used in this project to evaluate the performance of the different self-scheduling traffic control systems as well as a fixed time, cycle based approach for traffic control. This is done using different optimisation approaches and the open-source microscopic road traffic simulator, Simulation of Urban Mobility (SUMO)[8].

With ever more roads built and thus increasing traffic as well as advanced development being made in semi-autonomous vehicles, traffic light control is becoming a key challenge these years. Because of the resulting increase in traffic, congestion prevention and traffic flow management has become a major concern to transportation specialists and decision makers. To cope-up with these challenges and to meet the growing demand for traffic focused solutions, the urban transportation system needs effective solution methodologies. Thus, simulating and optimising traffic control algorithms for better accommodation and to deliver better solutions to urban mobility remains relevant up to day.

Thus, the project work focuses on the implementation and the performance analysis of different traffic controllers and compare them on three different sce-

narios. Additionally, an algorithm is proposed, which extends on the Self Organising Traffic Light (SOTL) algorithm [9] [4]. The main idea of the proposed algorithm is to hold onto the advantages of the SOTL algorithm and improve upon the shortcomings of SOTL observed during the project work. Therefore, an adaptive version of SOTL is introduced with our proposed *Adaptive Self Organising Traffic Light* (AdaSOTL) algorithm.

Equivalent to the approach for the other controllers, the performance of the proposed algorithm was examined in the same simulation environment, with optimized parameters, on the specified networks. These results were compared to the performance of the rest of the experiment results as well as the observations from [9].

Figure 1.1: Project structure

In Figure 1.1 the general approach during the project work is described and the structure of the technical report is organised as follows. At first, in Section 2 a brief review of prior research in adaptive traffic signal control and some related content is discussed. Next, in Section 3, the aims and approach of the project work is shown. The implementation and optimization approaches are described in Sections 4 and 5 respectively. In Sections 6 and 7 the experimental setup, networks used and the results that indicate the performance characteristics of the proposed approaches are presented. After that in Section 8 a discussion to the presented results is made. Finally in Sections 9 and 10 the work is concluded and some further research work is proposed.

2 Related Work

Several traffic light control approaches can be applied for organising traffic signals in an efficient way depending on the current traffic situation. Due to traffic light control in general being a wide discussed domain reaching far back in time, a classification of control methods can be applied [6]. Decentralized self-scheduling approaches like Platoon-based self-scheduling [9] are developed to further improve traffic flows and reduce congestion in real-life scenarios. The decentralized characteristic enables the controllers to not require any central controller and to not involve communication between local control units at intersections.

In addition to self-scheduling traffic light controllers, there are also self-organising approaches applied to control traffic lights [4]. In comparison to self-scheduling controllers which schedule or reschedule traffic light phases depending on the current traffic situation, self-organising controllers observe the traffic situation to organise traffic light switching at a specific time depending on specific traffic situations. This enables the possibility to react to special traffic situations or unexpected events, i.e. ambulance prioritisation [7].

The opportunity of traffic light control approaches to adapt their control behaviour to specific traffic situations are due to the use of hyperparameters. The efficient choice of their correct values in differing situations is crucial to the traffic control performance. The usage of optimisation to find specific hyperparameter values is common and widely spread. Depending on the characteristics of the traffic light controllers as well as the scenario complexity, less complex, like local search [1], or more complex, like metaheuristics [3], strategies are used to proceed optimisation processes effectively. In complex or specific cases, evolutionary or genetic algorithms are suitable for achieving efficient optimisation results in traffic light control hyperparameter search [2].

3 Goals of the Project and Approach

The key aspect of this project is to compare classical traffic light control meaning fixed-time, cycle based control with traffic light controls based on decentralised, agent-based methods.

Consequently, the following goals can be derived from these general statements, which were the working basis for the development of the project:

- Literature review about agent based simulation and traffic control
- Implementation of different traffic light controllers and testing on defined traffic scenarios
- Performance comparison between the different approaches
- Discussion on the gained results

Considering these goals, the traffic signal control problem focuses on maximizing the flow of traffic through specified road networks which are regulated by traffic lights. The main objective of these control systems is to assigning the green phases of each light such that the maximum number of vehicles traverse the network in a given time while considering safety and fairness constraints. While there are a variety of ways to measure the flow of the network, for this project the the average speed and the average waiting time of the vehicles for the given road network over a specific time period are used to measure performance. The faster the speed of the vehicles and the shorter the wait time, the better the flow is in the network.

Thus, these two measured values will be used as main indicators to compare implemented controllers and their results. Subsequently, this comparison should point out the strengths and weaknesses of the two approaches as well as give more insight in the different decentralised agent-based methods. To compare these approaches discrete simulation with optimization of the specific traffic light control parameters will be employed, where it is applicable.

4 Algorithms

This section describes the different algorithms used for the traffic signal problem. At first the fixed time cycle based approach is explained. Then, the implemented decentralized agent-based methods are presented. There the focus lies on the proposed AdaSOTL algorithm, with emphasis to the idea and the implementation strategy of the algorithm.

4.1 Fixed-time, cycle based control (FIX)

The first controller introduced is the classical traffic control strategy which assumes a cyclic operation of traffic lights, where each light moves through its sequence of phases in a fixed cycle time (t_{cyc}) that has an timed offset to the next traffic light. This offset is called phase shift (t_{shift}). Described in detail in [6], most conventional traffic control systems cycle timing plans are hand built and maintained, this means that designed time periods are sometimes given as statistical results based on the the historical data. As a result these traffic light control systems are somewhat static, i.e. traffic light signal changes in the designed time periods rather than following the real situation of vehicles passing intersections.

Thus, this control system typically requires a degree of stability in traffic flows over time to enable coordinating agents to build knowledge of current traffic flow patterns. To ensure the highest performance the traffic controller can provide in the certain scenarios, the cycle time and all phase shifts must be optimized.

4.2 Platoon-Based Self-Scheduling (PBSS)

As mentioned in [9], the basic idea behind the PBSS algorithm is to repeatedly collect data over a window of time between decision points, aggregate the data based on their proximity into groups (clusters), and then use these clusters to determine whether to extend the current phase, i.e., allocate the variable time, or transition to the next phase with the current phase being terminated. In addition, the length of time until the next decision point is computed and used for the next iteration of the algorithm. This loop is repeated indefinitely. There are three selection policies. They are, in order of priority:

1. Anticipated all clearing (AAC)
2. Platoon-based extension (PBE)

3. Platoon-based squeezing (PBS)

The AAC policy checks to see if there is an anticipated queue at the current time at the intersection for the road that is being serviced, i.e., has the green light. If so, it will try to extend the green light to service the vehicles in that queue. The PBE policy checks to see if there is a platoon on the road currently being serviced. If there is, then it tries to extend the current phase to service that platoon. The PBS policy checks to see if there is a platoon on the road that currently has a red light. If so, it determines if it can extend the current phase so that the transition to green for that road is timed to best serve that platoon.

For the final comparison four different variations of PBSS with different selection policies are utilised. The first strategy AAC only uses AAC, as the name suggest. Besides that, PBSSe uses only PBE and PBSSs uses only PBS, whereas PBSS employs all three policies.

4.3 Self Organising Traffic Lights (SOTL)

The SOTL algorithm, presented in [4], is another decentralized approach to control traffic lights. The basic idea is to integrate the number of cars approaching a red light. If this value (κ_i) reaches a certain threshold θ the light of the respective lane switches to green. In addition to this basic idea, a few constraints are added to the algorithm for stability and performance. As can be seen in the pseudo algorithm in Listing 4.1 line 3, a switching of greenphases only occurs, when a certain minimum green time (ϕ_{min}) has elapsed since the last switching operation. In this example, we use $\phi_{min} = 5$ seconds for the minimum green time. Furthermore, the algorithm tries to regulate, to some degree, the size of platoons passing through a crossway. If a lane has green and there is at least one car within a distance ω approaching the green light, then the traffic light will not switch until the car has passed through the crossway (listing 4.1 line 4). This is done to not cut off cars at the end of a platoon. However, for high density traffic, this constraint alone would become fatal, because if there are always some cars approaching the green light, it would never switch. Therefore, a second condition is applied, which allows switching of phases if there are more than μ cars in ω to cut platoons that are too large.

Algorithm 1: SOTL

repeat
  κ_i += $cars_{approachingRed}$ in ρ
  if $\phi_i \geq \phi_{min}$ **then**
    if not $(0 < cars_{approachingGreen}$ in $\omega < \mu)$ **then**
      if $kappa \geq \theta$ **then**
        $switchlight()$
        $\kappa_i = 0$
      end if
    end if
  end if
until simulation has ended

Listing 4.1: Self Organising Traffic Light (SOTL)

The SOTL algorithm is very intuitive and much more simple than the PBSS. If there are more cars approaching a red light, then they will get green faster, because the sum of integrated waiting times (κ_i) on their lane will reach the threshold level faster. This individual assignment of greenphases enables the SOTL controller to adapt to changes on the microscopic level.
On a macroscopic level, if the threshold θ is set to an adequate size which matches the current traffic load, then the loads from the different incoming lanes will be reflected in the overall distribution of the greenphase lengths (similar to the distribution of greenphases in the cycle based control).
However, there are a few weaknesses in the algorithm. For each specific traffic load scenario, there is one optimal threshold value θ e. g. the default value of $\theta = 42$ would be optimal for a light to medium traffic density. As we have seen in before in cycle based (Section 4.1): If the traffic gets denser, the greenphases should get longer. With a SOTL-controlled traffic light, the greenphases will get shorter in that case because the sum of integrated waiting times reaches the threshold value faster.

4.4 Adaptive Self Organising Traffic Lights (AdaSOTL)

Considering these weaknesses of the basic SOTL, lead to the proposal of an alternative algorithm to combine the strengths of the SOTL algorithm and overcome the observed shortcomings.

4.4.1 Idea

As mentioned above, when more cars are driving towards a red light of the crossway, the threshold is reached faster. This leads to green phases getting shorter when traffic load gets heavier. However, intuitively they should get longer. This in turn causes more yellow phases and crossing clearings, therefore less time for queue clearing. Thus, lost time for every crossway participant is the result. This observation is illustrated in figure 4.1, where the κ-dynamics (sum of integrated waiting time for one lane) over time in two different plots are shown. The first plot represents the dynamics with a fixed threshold value θ as used in SOTL, whereas the second plot shows the idea behind an adaptive value for the threshold θ.

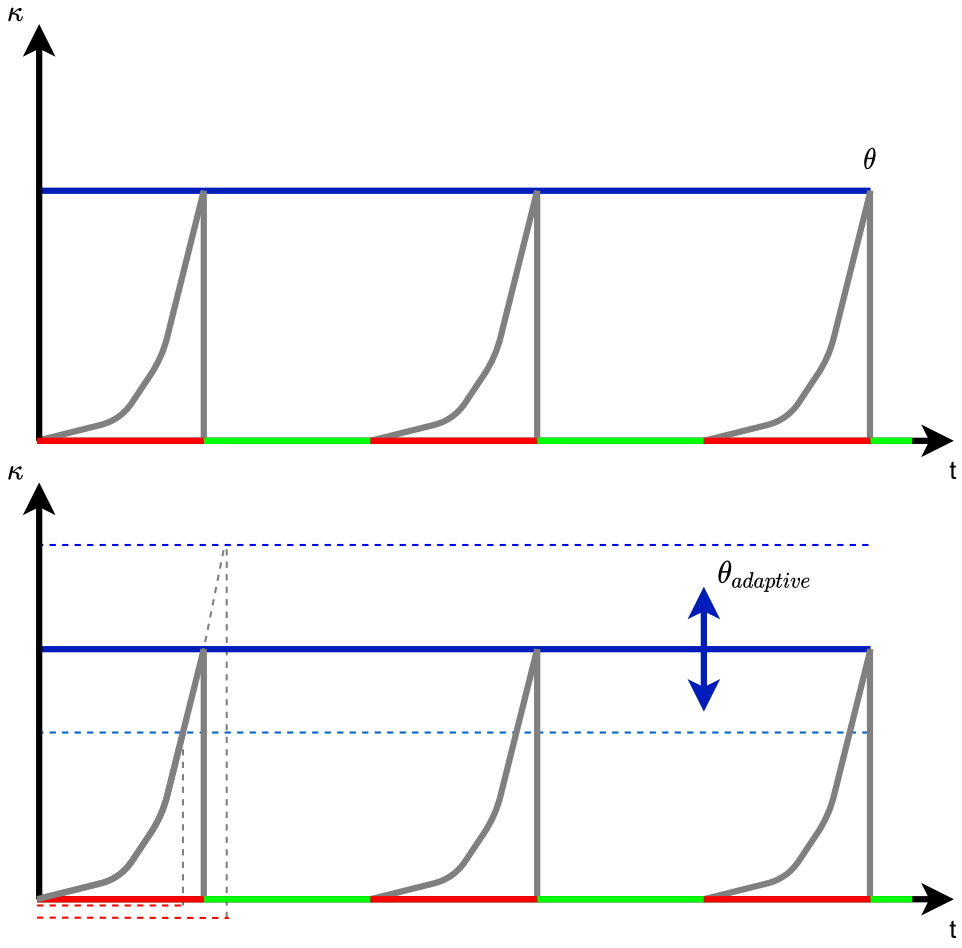


Figure 4.1: Illustration of the basic idea for an adaptive threshold

As an example if the κ value is rapidly growing it would reach the fixed threshold value very fast and a light switch happens. If a heavy traffic load is present this can occur after the defined minimum green time (t_{gmin}) which leads to constant light switches.

However, if this value could be made adaptive corresponding to the rise in κ, the fast light switches would be somewhat extended, as shown in the second plot. Likewise, if the κ increases slower, the threshold value would be lower to ensure a faster light switch if needed.

4.4.2 Implementation strategy

Thus, we propose an update to the existing algorithm, that implements this idea. Specifically, a SOTL version is presented, where the threshold value is not constant, but adapts itself to the present traffic load.

The local traffic density is proportional to the number of cars which are currently approaching a certain crossway. With the detectors already present in a SOTL-controlled traffic light, it is straightforward to get this information. Simply build the sum of cars approaching the crossway over all incoming lanes.

$$N = \sum_i n_i \tag{4.1}$$

Since this value is fluctuating quite a lot, we want to make it a little bit more stable. By calculating the running average of a time series we get an approximation of the mean in real time. We apply the running average to the numbers of cars approaching the crossway, so

$$N_{avg}(t) = (1 - D) * N_{avg}(t - 1) + D * N(t) \tag{4.2}$$

is a good representation of the current traffic density on a crossway, whereas $D \approx 0.1$ is the decay rate. If we multiply this value with a constant and use it as the new threshold for SOTL, the greenphases no longer get shorter, if the the number of cars increases, because the threshold would increase equally. Now to model the fact, that the greenphases should get longer, as traffic increases, the threshold must increase at a faster rate than the actual traffic. This can be achieved by potentiating N_{avg}. Our new threshold, which is recalculated every simulation step to adapt to the situation, is calculated by

$$\theta = {N_{avg}}^{\beta} * \alpha \tag{4.3}$$

where:

θ = Threshold value for waiting cars
N_{avg} = the running average of the cars approaching the crossway
α = proportionality constant to scale threshold with increasing traffic
β = exponent to ensure increasing greentime length when traffic rises

With regards to this implementation strategy, the following pseudo algorithm is proposed in listing 4.2. In this code, the highlighted lines in red (ln. 2-4) indicate the presented changes to the SOTL pseudo algorithm (listing 4.1).

Algorithm 2: AdaSOTL

Require: D, α, β
1: **repeat**
2: $N = \sum_{Lanes} cars_{approachingCrossway}$
3: $N_{avg} = (1 - D) * N_{avg} + DN$
4: $\theta = \alpha * N_{(avg)}^{\beta}$
5: **for** Red lanes i **do**
6: **if** $\phi_i \geq \phi_{min}$ **then**
7: κ_i += $cars_{approachingRed}$ in ρ
8: **if** not $(0 < cars_{approachingGreen}$ in $\omega < \mu)$ **then**
9: **if** $\kappa_i \geq \theta$ **then**
10: $switchlight()$
11: $\kappa_i = 0$
12: **end if**
13: **end if**
14: **end if**
15: **end for**
16: **until** simulation has ended

Listing 4.2: Adaptive self organising traffic light (AdaSOTL)

5 Optimization

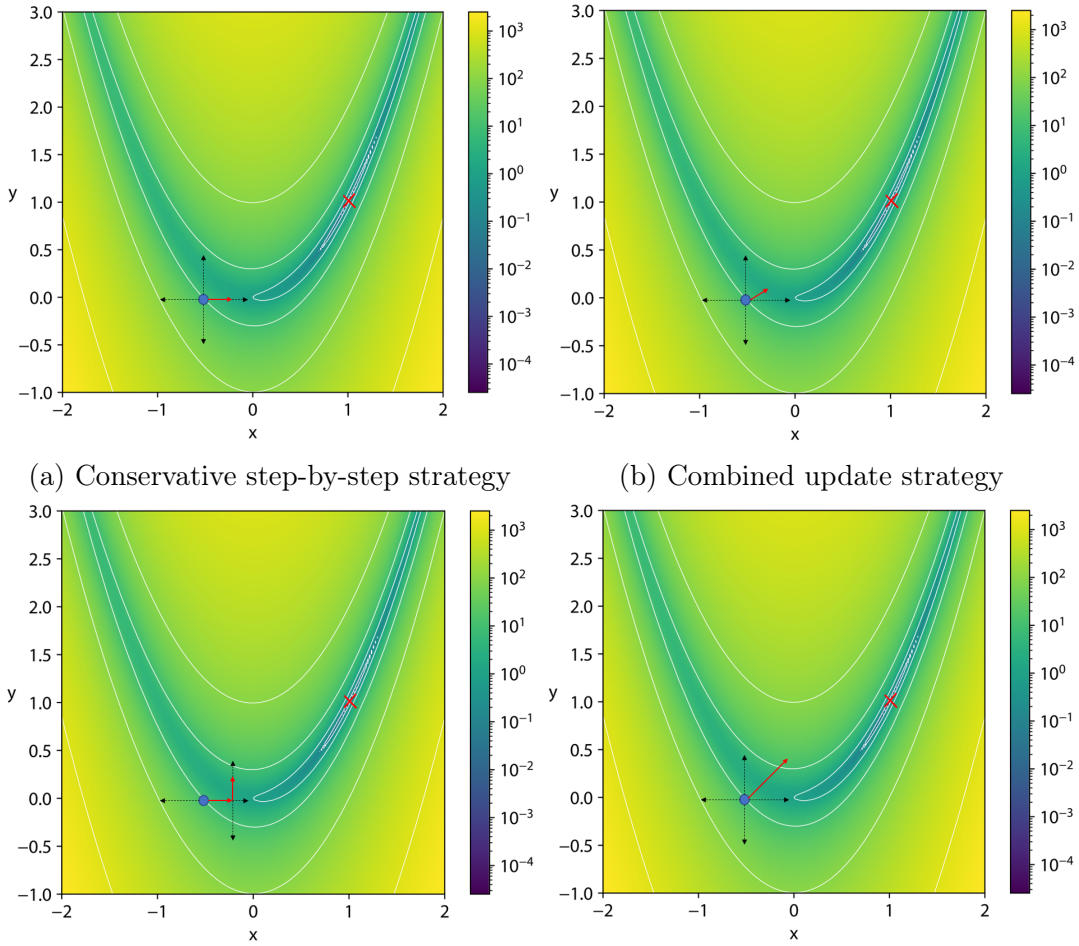
Considering the previously introduced traffic lights control approaches with its individual hyperparameters, finding optimal values for these customizable parameters is essential for high performance in the specific traffic scenarios [1]. Dependent on the objective function being optimized and therefore the problem domain's complexity, the optima can be deterministic or non-deterministic. The problem domain of traffic lights control including traffic simulation considered in this work defines a search space being dependent on many system parameters. This leads to the lack of a mathematically describable objective function and results in using non-linear, multidimensional optimization strategies. In this chapter, the implementations of the Hill Climbing search algorithm as a local search algorithm as well as the $(\mu/\mu, \lambda)$-σ-Self Adaptation Evolution Strategy as a metaheuristic [3] are described. The remarks on the objective function are stated in 6.2 after the scenario descriptions.

5.1 Hill Climbing search algorithm

Our implementation of the Hill Climbing search algorithm follows the basic idea of the local search. Starting the search at random values within the specified search space, we evaluate the objective function through increasing and decreasing the parameters by suitable step sizes. Dependent on the evaluation results and the specified optimization goal (maximization or minimization), we define the search direction and update the parameters by calculating the differences of the achieved objective values between the original and the increased/decreased parameters. The optimization process is finished, if either the differences between the objective values of the current and previous iteration are below a certain threshold ϵ or if the maximum number of iterations is reached. The search of the optimization direction in each iteration as well as the parameter update process can be carried out in different manners. We implemented four different strategies to cover these procedures. Their individual search behaviours are exemplary visualized on the Rosenbrock function in Fig. 5.1.

Conservative step-by-step strategy

Following a conservative behaviour, our first search strategy evaluates the objective function for each dimension in positive and negative direction with by step sizes increased/decreased parameter values. The achieved objective values are compared to the objective value achieved by the original parameters.



(a) Conservative step-by-step strategy

(b) Combined update strategy

(c) Greedy iterative steps strategy
Red cross: optimum
Black dotted arrows: search directions

(d) Discrete updates strategy
Blue circle: current parameters
Red arrows: actual updates

Figure 5.1: Exemplary search and update behaviours of the suggested strategies on the Rosenbrock function

Thus, the parameter update for each dimension i is performed by

$$param_i = \begin{cases} param_i + difference_i * stepSize_i, & i = best \\ param_i, & otherwise \end{cases} \tag{5.1}$$

The optimization step is performed in the direction which results in the biggest positive update of the objective value for all dimensions. Using this strategy, it is guaranteed to converge to the optimum due to only performing optimization steps towards the optimum. Nevertheless, the optimization efficiency is low due

to only considering one objective evaluation result at performing the update step. This can lead to a very high computational effort with increasing search space dimensionality.

Combined update strategy

To take more objective function evaluations for the parameter update into account, we suggest a strategy performing optimization steps in multiple dimensions contemporaneous. Similar to the previously described strategy, the objective function is evaluated with step size increased/decreased parameters for all dimensions. Then, the $difference_i$ factor in each dimension i is derived by

$$difference_i = \frac{objectFunc(incParams_i) - objectFunc(decParams_i)}{2} \quad (5.2)$$

where $objectFunc()$ is the objective function evaluation and $incParams_i$/ $decParams_i$ are the parameters being increased/decreased by the specific step size in the dimension i. First, the update in each dimension is performed in the direction which leads to the biggest positive objective value change. Second, performing optimization steps in multiple dimensions contemporaneously increase the efficiency of objective function evaluations and enable optimization steps across dimensions (e.g. diagonal steps in two dimensional search space). Despite the increase in efficiency, the bundled optimization step can also lead to a negative reaction of the objective value at reevaluating the objective function with the updated parameters. This is reasoned by the fact, that the individual positive changes to the objective value due to the dimension exclusive updates do not guarantee a positive change to the objective value by using the updated parameters combined. Therefore, by using this strategy, we also accept optimization steps which result in negative changes to the objective value in order to possibly converge faster to the optimum.

Greedy iterative steps strategy

As a third strategy we suppose a greedy optimization behaviour. Thus, we iterate over the dimensions and evaluate the objective function with increased/decreased parameters. In comparison to the combined update strategy, where every performed function evaluation is considered for the optimization step, an update is performed immediately after achieving a positive change to the objective value. For example, if the first parameter gets increased/decreased

and achieves a better objective value in at least one direction, the parameter is immediately updated following the upper update rule in Eq. 5.1 (here i = best direction, increase/decrease respectively). The next evaluation considering the next dimension is already performed with the updated parameters. Following this strategy, both advantages of the previously described strategies are combined. Nevertheless, the high parameter update frequency can lead the optimization behaviour to inefficient paths/areas in the search space. Additionally, not accepting worse objective values can hinder the optimization process to converge to the optimum in regards of noise and extremely multi-modal search spaces.

Discrete updates strategy

Due to the existence of traffic light controlling approaches having discrete hyperparameters, like θ of SOTL or the phase shifts of the cycle based approach, the continuous updates of these hyperparameters performed in all previously described strategies decelerate the optimization process. Therefore, we suggest a forth strategy which chooses the direction of the optimization step like one of the previous strategies but updates the parameters only by adding/subtracting the constant step sizes. To still guarantee adaptivity in the optimization process, the step sizes are halved each time the threshold ϵ is reached to perform smaller search steps.

5.2 $(\mu/\mu, \lambda)$-σ-Self Adaptation Evolution Strategy

Due to the complex and realistic problem domain of traffic light control, an evolution strategy, a metaheuristic respectively, is also considered for optimization [3]. The $(\mu/\mu, \lambda)$-σ-Self Adaptation Evolution Strategy is implemented based on the algorithmic structure proposed in [2]. Our basic implementation is shown in Listing 5.1.

Algorithm 3: $(\mu/\mu, \lambda)$-σ-SA ES

Require: $y, \mu, \lambda, \sigma, \sigma_{stop}, \tau, maxGenerations$

repeat
  for l = 1:λ **do**
    $\sigma_l = \sigma e^{\tau \mathcal{N}_l(0,1)}$
    $y_l = y + \sigma_l \mathcal{N}(1, 0)$
    $f_l = f(y_l)$
    $o_l = (f_l, y_l, \sigma_l)$
  end for
  sort$(o_1, ..., o_\lambda)$
  $y = (\sum_{k=1}^{\mu} o_k)/\mu$
  $\sigma = (\sum_{k=1}^{\mu} \sigma_k)/\mu$
until $\sigma < \sigma_{stop}$ or $maxGenerations$

Listing 5.1: $(\mu/\mu, \lambda)$-σ-Self Adaptation Evolution Strategy

Starting with a random set of parameters, the initial parent y respectively, λ parameter variations (offsprings) are generated by mutating the parent. The mutational impact is controlled by the mutation strength σ. Every offspring is evaluated on the objective function and sorted in ascending/descending order for minimization/maximization. The μ best offsprings are recombined by creating their centroid. The recombinant is declared as the new parent and the recombined σ is the new mutation strength for the next generation.
Due to the self-adapting mutation strength, the parameter updates which are also the performed optimization steps are adapted automatically. Nevertheless, to achieve efficient optimization steps per generation, a certain amount of offsprings λ must be generated to create efficient recombinants. Due to the time consuming simulations of traffic scenarios, this characteristic can be a usage bottleneck of the evolution strategy. Additionally, the parameters are mutated continuously which can decelerate the optimization, as described previously.

6 Experiments

To verify a correct implementation of the control systems, they are tested on the same networks used in [9]. Thus, the experiments are done on the 1x5 Arterial Network and the 2x3 Grid Network. The networks have five and six intersections, respectively, and all roads are one-way. Each road starts at either an intersection or an entry point into the network, and each road ends at either an intersection or an exit point from the network.

For each network, the car's maximum speed is 10 m/s and the distance between nodes (L) is identical with $L = 500$ meters. On each road, a traffic detector to detect the incoming traffic is located at $L_{det} = L - 50$ meters from each intersection. The minimum green time is $\tau_{gmin} = 5$ seconds for the Platoon-based self-scheduling approaches and $\tau_{gmin} = 20$ for remaining controllers, the maximum green time is $\tau_{gmax} = 55$ seconds and the yellow time is $\tau_y = 5$ seconds. Every optimization experiment is carried out with 50 iterations and 5 replications. The final objective value is achieved by calculating the average over every objective value gained from replications.

6.1 Traffic Networks

6.1.1 1x5 Arterial Network

The first network is an arterial road with five cross streets. Incoming traffic is divided among the roads with the proportions shown in Figure 6.1. No turns will occur at any intersection except **O**, where traffic from the cross street turns onto the artery with probability $r_t/(r_s + r_t)$. Initially the probability of r_t is zero. The total simulation period is one hour, and every twenty minutes the turning proportion increases as $r_t = r_t + \Delta r_t$ with $r_s + r_t = \frac{5}{16}$.

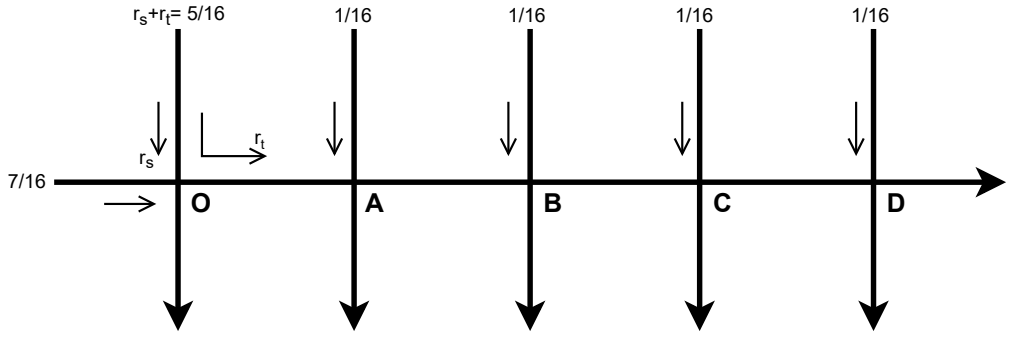


Figure 6.1: 1x5 Aterial Network

6.1.2 2x3 Grid Network

The second network, shown in Figure 6.2, is a grid with six intersections. Compared to the arterial network, there are more possible routes a vehicle can take. For the experiment eight routes are used. These consist of the straight routes on each road, which have minor traffic flows which generate 1/12 of total traffic each, as well as three routes with major traffic flows. Like with the arterial network the total simulation time is one hour, and for each twenty minute period, a different major route (**a**, **b**, **c**) is generating 7/12 of the total traffic.

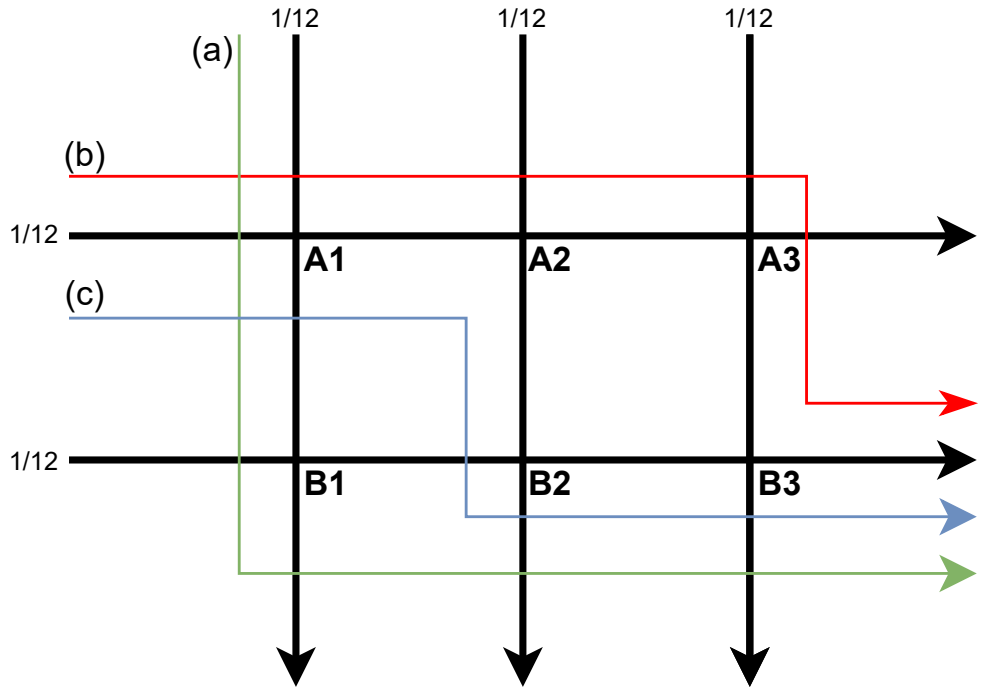


Figure 6.2: 2x3 Grid Network

6.1.3 2x3 Grid Network with uniform flow

As an additional method of comparing the cycle based and the agent based approaches, the 2x3 Grid Network is used with a uniform flow during the simulation. Thus, there are no flow switches and flow **(a)** (as seen in Figure 6.2) is used for the whole simulation time. This is done because both used networks (1x5 and 2x3) employ flow switches during the simulation time, where the cycle based approach needs more time to adjust to them. Therefore, a network with

uniform flow gives additional insight in the behaviour of the two approaches and how the fixed cycle based method performs during this particular scenario.

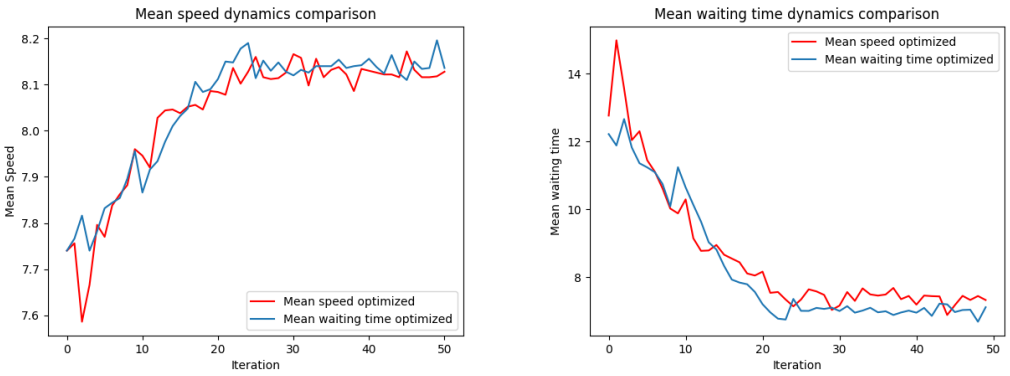
6.2 Objective function

To ensure comparability to the experiments of [9], the mean speed ($V_{n,all}$) and the mean waiting time ($T_{w,all}$) are possible key performance indices (KPIs) for optimization. Our approach considers a mono-objective optimization. Both KPIs exhibit similar characteristics being continuous and bounded. $V_{n,all}$ is bounded by achieving values between 0 and 10 m/s. The values for $V_{n,all}$ are extracted from the file *statistics.xml*. $T_{w,all}$ is bounded in range 0 seconds and maximum waiting time possibly achievable dependent on maximum green time. [9] calculate $T_{w,all}$ by collecting the mean waiting time per junction and calculate the average over all junctions. Since we only have access to the trip informations per car after the simulation, the $T_{w,all}$ calculation in the paper must be approximated. Therefore, we calculate $T_{w,all}$ with information of all cars by

$$T_{w,all} = \sum_{i=1}^{all} T_{w,i} / \sum_{i=1}^{all} numJunctions_i \tag{6.1}$$

to correctly weigh cars dependent on how many junctions they passed on their route.

To decide between the usage of $V_{n,all}$ and $T_{w,all}$ as the objective function, a comparison of the optimization dynamics of both objective functions is performed on the 1x5 Arterial Network with $\Delta r_t = 2/16$. The comparison is visualized in Fig. 6.3.



(a) $V_{n,all}$ comparison

(b) $T_{w,all}$ dynamics comparison

Figure 6.3: Comparison of $V_{n,all}$ and $T_{w,all}$ as objective functions for optimization on the 1x5 Arterial Network with $\Delta r_t = 2/16$

As the analysis of the optimization dynamics only show insignificant differences, one could suppose an equality of performance for $V_{n,all}$ and $T_{w,all}$ as objective functions. To reinforce this hypothesis, the Pearson correlation coefficient for $V_{n,all}$ and $T_{w,all}$ is calculated over every optimization done on the 1x5 Arterial Network. The resulting coefficient of -0.91 meaning a strongly negative correlation between both possible objective functions can lead to the acceptance of the hypothesis.

In our opinion, $V_{n,all}$ implicitly also provides information about how well the traffic flows and how fast vehicles pass through the considered scenario incl. waiting times. Therefore, we decided to use $V_{n,all}$ as the objective function for our optimizations.

7 Results

The following chapter shows the simulation results of how the implemented controllers performed on the previously described networks with different traffic load scenarios. For all cases, the mean speed ($V_{n,all}$) and the average waiting time per vehicle per crossway ($T_{w,all}$) have been measured.

7.1 1x5 Arterial Network

The results of the arterial network are shown below in table 7.1. The AdaSOTL controller has the highest average speeds and lowest mean waiting times in all scenarios.

Table 7.1: Results of the 1x5 Grid

Controller	$\Delta r_t = 0$		$\Delta r_t = 1/16$		$\Delta r_t = 2/16$	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
FIX	8.00	7.92	7.97	8.17	7.95	7.79
SOTL	7.83	17.59	7.83	19.10	7.63	26.26
AAC	7.83	9.35	7.63	10.32	7.58	10.53
PBSSe	7.80	9.51	7.60	11.28	7.60	11.29
PBSSs	7.72	9.58	7.64	10.21	7.60	10.25
PBSS	7.84	9.07	7.61	10.62	7.55	10.99
AdaSOTL	**8.29**	**6.51**	**8.19**	**7.52**	**8.17**	**7.18**

7.2 2x3 Grid Network

Table 7.2 shows the results of the 2x3 grid network. Again, AdaSOTL performs best through all traffic load scenarios. It has to be mentioned, that there is most likely a little bug in the platoon based extension policy (PBE) since there is a unusually high spike in the mean waiting time on the grid network with 1500 vehicles. However, the information on the algorithm given by [9] has some ambiguity to it, so it remains to be fixed later.

Table 7.2: Results of the 2x3 Grid

Controller	900 Vehicles		1200 Vehicles		1500 Vehicles	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
FIX	7.93	13.32	7.05	28.96	7.02	43.75
SOTL	7.14	20.53	5.68	42.12	5,03	53.32
AAC	8.08	**6.48**	7.70	9.40	6.96	14.82
PBSSe	8.11	7.16	7.66	13.75	6.45	72.26
PBSSs	8.03	6.90	7.71	8.51	7.03	11.62
PBSS	8.11	6.79	7.75	10.63	7.03	21.50
AdaSOTL	**8.22**	6.77	**8.04**	**7.56**	**7.80**	**9.05**

7.3 2x3 Grid Network without flow switches

When traffic density stays constant on the GridNetwork, cycle based becomes the best control method. This is due to the optimization process, where the phaseshifts have been optimized to maximize the flow through the network. However, the behaviour of real traffic is very unlikely to be that predictable, so these results are to be enjoyed with caution. The gap between cycle based and AdaSOTL gets increasingly smaller, as the traffic increases.

Table 7.3: Results of the 2x3 Grid without flow switches

Controller	900 Vehicles		1200 Vehicles		1500 Vehicles	
	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$	$V_{n,all}$	$T_{w,all}$
FIX	**8.61**	**3.35**	**8.38**	**4.38**	**7.95**	9.82
SOTL	6.82	27.41	5.09	43.35	4.92	56.37
AdaSOTL	8.35	5.81	8.14	7.34	7.87	**9.13**

8 Discussion

Since it is not that straightforward to determine from these results which traffic light controller is best, we qualitatively compare the controller against each other to get a better impression of where the strengths and weaknesses of the controllers lie. For this purpose, we defined four requirements a good traffic light controller has to meet:

1. Adapt to changes on macroscopic level
2. Adapt to changes on microscopic level
3. Efficiently structure and guide traffic through the network
4. Easy implementation

In general, there are two levels of variation occurring in the traffic. On a macroscopic level, the flows which have been specified by us can vary in their routes and density. For example, in the arterial network, the turn probability of cars merging in the arterial increases every twenty minutes and in the Grid Network, the main route from northwest to southeast uses different paths through the grid.

On the other hand, there are changes on a microscopic level. These happen at the sources of the network, where we assume that vehicles are coming from an uncontrolled (no traffic lights) street. In the simulation, these microscopic changes are caused by exponentially distributed inter-arrival times at the sources of the networks.

A good traffic light control system would be able to adapt to changes on both the macroscopic level and the microscopic level, as well as structuring the traffic in a reasonable manner (e. g. platoons) and guiding it through the network. As a fourth criteria, we added the implementation difficulty, because in reality, each crossway is different from the next one, and the controller would need to be adapted to the given situation. An algorithm, which is easy to understand and to implement, is also easier to change and therefore costs less time and is less prone to error. The following Table 8.1 shows a qualitative comparison of the traffic light control algorithms based on these requirements.

Algorithm	macroscopic	microscopic	guiding of traffic	Difficulty	total
FIXED	●	●●	●●●	●●	8
SOTL	●	●	●	●●●	6
PBSS	●●●	●●●	●●	●	9
AdaSOTL	●●	●●●	●●	●●	9

Table 8.1: Qualitative comparison of the algorithms

8.1 Cycle based

The fixed time, cycle based controller is best at structuring the traffic in a reasonable manner and guiding it through the network. This is because it is the only controller with global information available. More specifically, the phase offsets between traffic lights have been optimised to best serve the incoming cars. The algorithm performs best, when there are no changes on the macroscopic level (as in arterial network with $\delta_{rt} = 0$ and Grid Network without flow switches), because cycle based traffic lights cannot adapt to these changes. However, in reality this is never the case. On the Grid Network, when the main flow changes suddenly every twenty minutes, vehicles from the previous main flow are still present in the network and get stuck, because the traffic lights are programmed to best serve the current flow. This reduces the performance of the cycle based controller drastically. In reality, this effect would be even greater, because real traffic is not as predictable as in the simulation and changes occur gradually and continuously so that there is never a constant load to which the system can be optimised to.

8.2 SOTL

The SOTL algorithm performs worst in all scenarios. However, after giving the algorithm a chance to adapt to the specific traffic loads during optimization, the performance was significantly better than in the paper, which served as benchmark for the project. Advantages and disadvantages of the SOTL are explained in chapter 4.

8.3 PBSS

The group of platoon based algorithms performs well on all networks and traffic loads. They even outperform cycle based control in the grid network with flow

switches. This is because the network is to complex to optimize the cycle based controller in a reasonable manner.

It is also important to mention, that for a direct comparison of the algorithms, PBSS has an unfair disadvantage, because all the other control methods had their parameters specifically adapted to the respective scenario, while PBSS parameters were taken "as is". The optimised Hyperparameters used in the different scenarios are listed in Appendix 10.

8.4 AdaSOTL

In our experiments, the AdaSOTL algorithm has the best overall performance. Only on the grid network without flow switches, the cycle based controller is slightly better. This shows, that - as expected - AdaSOTL with the adaptive threshold is able to adapt to both levels of variation.

As mentioned before, AdaSOTL, as well as the other controllers have an unfair advantage against the PBSS algorithm group, but we still believe, that it deserves recognition for the results that it achieved and that further investigation on this algorithm is reasonable.

9 Conclusion

To conclude, it can be said that in our experiments the decentralised agent-based approach performs better than the cycle based approach. Especially in regards to the newly proposed AdaSOTL algorithm. We have seen, that our AdaSOTL has the best overall performance of all implemented controllers. Only on the 2x3 Grid Network without flow switches, the cycle based controller is slightly better. PBSS performs good as well and is always very close behind AdaSOTL regarding waiting times and mean speed. In the qualitative comparison, both AdaSOTL and PBSS scored 9 points. AdaSOTL is easier to implement but not yet as good as PBSS in adaptability to macroscopic changes, because for AdaSOTL to be better than PBSS the Hyperparameters had to be adapted to each scenario. This is also the reason why we concluded that unless, we manage to find one set of parameters for the AdaSOTL which works for all of the scenarios, PBSS will remain the best algorithm. However, if we do find them and the algorithm can still compete with PBSS regarding performance, we could say that, given the fact that AdaSOTL approach is much more simple, it would be the better option.

10 Future Work

- To improve AdaSOTL, it is planned to investigate other possibilities on how the adaptive threshold value can be calculated. The equation 4.3 yields quite different hyperparameters for the different scenarios (see Appendix 10). In theory, a good method has been found, if the optimization process yields the same or similar hyperparameters for all scenarios and performance is still good.
- Ongoing experiments look promising and when the upgraded AdaSOTL is ready, it is planned to build a model of a real network and simulate it with the upgraded AdaSOTL.
- In [4] a similar idea was mentioned. The authors noticed as well, that there is a monotonic correlation between the traffic density and the optimal threshold value and verbally proposed the idea of implementing the SOTL algorithm with a variable threshold. Following that lead, we have to find out if that idea already exists elsewhere.
- If our idea still contributes something new to the topic and experiments with the realistic scenario yield good results we want to write a scientific article about it and publish it on relevant platforms.

Bibliography

[1] Mohammed Azmi Al-Betar et al. "Adaptive $\beta-$hill climbing for optimization". en. In: *Soft Computing 23* (2019), pp. 13489–13512.

[2] Hans-Georg Beyer and Hans-Paul Schwefel. "Evolution strategies: A comprehensive introduction". en. In: *Natural Computing 1* (2002), pp. 3–52.

[3] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. "A survey on optimization metaheuristics". en. In: *Information Sciences 237* (2013), pp. 82–117.

[4] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. "Self-Organizing Traffic Lights: A Realistic Simulation". In: *Advances in Applied Self-Organizing Systems* (Oct. 2006). ISSN: 978-1-4471-5112-8. DOI: 10.1007/978-1-84628-982-8_3.

[5] European Court of Auditors. *Sustainable urban mobility in the EU : no substantial improvement is possible without Member States' commitment*. eng. Special report No ... (European Court of Auditors. Online). LU: Publications Office, 2020. URL: https://data.europa.eu/doi/10.2865/517496.

[6] Stefan Lämmer. "Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straßennetzwerken". de. PhD thesis. Technischen Universit¨at Dresden, Apr. 2007.

[7] Marcin Lewandowski, Bartłomiej Płaczek, and Marcin Bernas. "Self-organizing Traffic Signal Control with Prioritization Strategy Aided by Vehicular Sensor Network". en. In: *Computer Information Systems and Industrial Management*. Ed. by Khalid Saeed, Władysław Homenda, and Rituparna Chaki. Vol. 10244. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 536–547. ISBN: 978-3-319-59104-9. DOI: 10.1007/978-3-319-59105-6_46. URL: https://link.springer.com/10.1007/978-3-319-59105-6_46 (visited on 01/17/2022).

[8] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation using SUMO". In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL: https://elib.dlr.de/124092/.

[9] Xaio-Feng Xie et al. "Self-Scheduling Agents for Real-Time Traffic Signal Control". en. In: *Technical Report, CMU-RI-TR-11-06* (), p. 18.

Appendix

The following tables contain the results of the optimization experiments. For all of the gained results these hyperparameters were used.

Table 10.1: Optimized hyperparameters of SOTL algorithm

	1x5 Network			2x3 Network			2x3 Network fixed		
	$\Delta r_t = 0$	$\Delta r_t = 1/16$	$\Delta r_t = 2/16$	900	1200	1500	900	1200	1500
θ	24	24	24	17	29	30	28	30	34

Table 10.2: Optimized hyperparameters of AdaSOTL algorithm

	1x5 Network			2x3 Network			2x3 Network fixed		
	$\Delta r_t = 0$	$\Delta r_t = 1/16$	$\Delta r_t = 2/16$	900	1200	1500	900	1200	1500
α	1.39	3.49	2.69	3.21	3.94	4.50	3.39	3.49	3.09
β	1.79	1.39	1.55	1.32	1.41	1.48	1.55	1.61	1.59

Table 10.3: Optimized hyperparameters of Cycle based algorithm

	1x5 Network			2x3 Network			2x3 Network fixed		
	$\Delta r_t = 0$	$\Delta r_t = 1/16$	$\Delta r_t = 2/16$	900	1200	1500	900	1200	1500
ct_{factor}	0.81	0.81	0.81	0.86	0.96	0.8	0.86	0.86	0.75
t_{shift^1}	64	64	64	54	86	48	80	36	74
t_{shift^2}	47	47	47	55	57	55	49	53	25
t_{shift^3}	87	87	87	99	89	93	99	101	53
t_{shift^4}	70	70	70	66	82	86	52	56	84
t_{shift^5}	-	-	-	157	135	125	107	155	135

Statutory Declaration

We declare that we have authored this thesis independently, that we have not used other than the declared sources / resources, and that we have explicitly marked all material which has been quoted either literally or by content from the used sources. The thesis was not examined before, nor has it been published.

February 22, 2022 | Marcus Bentele, Georg Fessler, Thomas Rosenberger